Reliable Video over Software-Defined Networking (RVSDN)

Harold Owens II
Indiana U.-Purdue U. Indianapolis
Dept. of Comp. and Information Sci.
Indianapolis, IN USA
Email: owensh@cs.iupui.edu

Arjan Durresi
Indiana U.-Purdue U. Indianapolis
Dept. of Comp. and Information Sci.
Indianapolis, IN USA
Email: durresi@cs.iupui.edu

Raj Jain
Washington University
Depart. of Computer Sci. and Eng.
St. Louis, Missouri USA
Email: jain@cse.wustl.edu

***Abstract*—Ensuring end-to-end quality-of-service for video applications requires the network to choose the most feasible path in terms of bandwidth, delay and jitter. Quality of service can only be ensured if the paths are *reliable* - perform to specification per request. This paper makes four contributions to research. First, it presents Reliable Video over Software-Defined Networking (RVSDN) which builds upon previous work of Video over Software-Defined Networking (VSDN) to address the issue of finding the most reliable path(s) through the network for video applications. Second, it presents the design and implementation of RVSDN. Third, it presents the experience of integrating RVSDN into ns-3 which is a network simulator used by the research community to simulate and model computer networks. Finally, it presents the results of RVSDN in terms of the number of requests serviced by the network architecture. RVSDN is able to service 31 times more requests than VSDN and MPLS explicit routing when the reliability constraint is 0.995 or greater using aggregation of reliability across network paths.**

I. Introduction

Video demands across the Internet are projected to grow to nearly 69 percent of Internet traffic by 2017 [1]. The increase in video demand is caused by hardware (*e.g.*, smart TVs, tablets and smart phones) and software (*e.g.*, Facebook, YouTube, Netflix and HuLu). Video applications such as video-on-demand (VOD) and telesurgery are pushing the current network infrastructure and protocols to the limits. These factors require that video traffic have a certain level of quality-of-service (QoS) from the network. The QoS that the network provides to video applications can be required bandwidth and minimal delay and jitter. Currently, there are QoS frameworks (*e.g.*, differentiated services, integrated services and MPLS) that provide QoS for real-time application like video but each has its limitations [2]. The QoS frameworks and the Internet were not developed for the great demands of video applications of today. New approaches to handle QoS, security, reliability and wireless technologies within the Internet are needed [3].

Supporting today's video applications require us to rethink how the network should provide end-to-end QoS guarantees. Currently, network QoS frameworks consider network bandwidth, delay and jitter. Although these attributes are important to real-time applications like video, meeting these constraints do not address reliability of network paths or build the confidence of network operators about the path selection process. In this paper, reliability is the ability of the network to perform to specification [4] per request. This paper builds on our past work [5] and investigates the ability of the network to select reliable network paths. Network frameworks may use multi-path selections to decrease the probability of a failure, handle failures more gracefully or increase bandwidth capacity [6]. Multi-path selection allows the network to load balance network traffic and to provide fail-over in case the primary path fails. Multi-path selection does not address the reliability issue of video applications such as remote surgery, robotic packets or interactive video.

Constraint-based routing or multiple path selection can fail to provide end-to-end quality of service for video applications. For example, if the QoS requirement for a video application is 0.95 reliability, 1.5Mbps bandwidth, 100ms delay, and 20ms jitter, current QoS frameworks like integrated services find a `Path1` which meets the required constraints (*i.e.*, bandwidth, delay and jitter) but will be unable to meet reliability constraint because supporting reliability constraint is not built into architecture design. The network operator may configure MLPS fail-over links to address the issue of reliability. Although, availability of the network is increased using fail-over links, fail-over links will not necessarily provide the required reliability for video application because MPLS explicit routing paths are static. If network operator configures MPLS fail-over as `Path1` with reliability of .75 and `Path2` with reliability of .75, the overall reliability of the disjointed paths would be $(1 - (1 - .75) \times (1 - .75)) = 0.9375$ which does not meet the reliability of requirement 0.95. Furthermore, the static routes of MPLS could not dynamically select a combination of reliable paths to service real-time applications like video.

Meeting reliability requirement for video applications requires the path selection process to consider more than a single path even if path meets reliability requirement before a failure. It requires the path selection algorithms to dynamically consider the combination of multiple paths' bandwidth, delay, jitter and reliability constraints. The network QoS frameworks currently do not address the issue of reliability which is the main idea of this paper.

Therefore, this paper presents experience and results of integrating reliability support into the Video over Software-Defined Networking (VSDN) architecture which ensures end-to-end QoS for video applications. The main contributions of this paper are:

- It presents *Reliable Video over Software-Defined Networking (RVSDN)*, which is an architecture that builds on previous work [5] to ensure end-to-end quality of service for video applications;
- It presents the results of implementing RVSDN into the NS3 simulator [7]; and
- It presents an empirical study that evaluates RVSDN runtime performance in terms of number of requests serviced per reliability constraint request.

Paper organization. The remainder of this paper is organized as follows: Section II motivates the need for Reliable Video over Software-Defined Networking (RVSDN); Section III discusses the design and implementation of RVSDN; Section IV presents results of simulating RSVDN in a simulator and interpretation of the results; Section V compares RVSDN to related works; and Section VI provides concluding remarks.

II. Overview of IntServ QOS Model and Current Implementation of Video over Software-Defined Networking (VSDN)

This section briefly discusses IntServ architecture. It also discusses VSDN's current implementation and limitations in ensuring end-to-end QoS for video applications.

A. Overview of Integrated Services (IntServ)

IntServ architecture uses a reservation protocol to configure end-to-end QoS over IP networks. The necessary resources (*e.g.*, computation and storage) are reserved at each router along the packet's path. A `PATH` message is sent by the sender to receiver. The `PATH` message follows the exact path as the IP packet; it can not be sent along a different path. The `PATH` message leaves states along the packet's path at each router (*i.e.*, hop). The receiver responds to sender with a `RESV` which is used to reserve the resources along the packet's path after receiving and accepting the `PATH` message. The `PATH` and `RESV` message configures soft states (*e.g.*, rate, max queue size, peak data rate and minimal packet size) at each router. These metrics are used to ensure packets of a specific flow receive the guaranteed QoS. The software states at each router must be refreshed periodically to ensure sessions do not timeout.

IntServ advantages include software state adaptability, ability of receiver to initiate reservation and the ability for routers to merge reservations [2]. One key disadvantage of IntServ is the inability to select a different QoS path that differs from the routing protocols (*e.g.*, IS-IS, OSPF and RIPv2). For example, in figure 1 let us assume that the best path for video in terms of bandwidth, delay and jitter

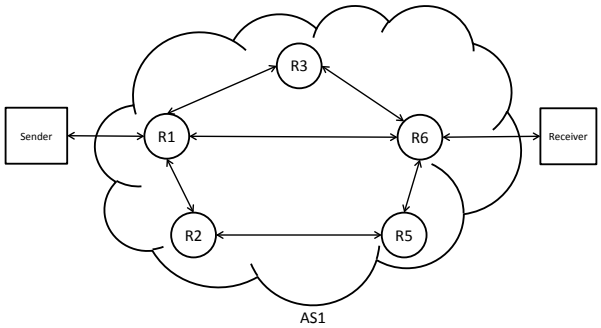

Figure 1: Simple Network Topology with a Sender and a Receiver

is `R1 - R2 - R5 - R6`. If the routers in autonomous system one (AS1) are running OSPF, the shortest path would be calculated to be `R1 - R6`. Therefore, video packets in IntServ from sender to receiver will traverse `R1 - R6` which is two hops.

In this case, IntServ would ensure quality of service at routers `R1` and `R6` but as previously stated, the best path is `R1 - R2 - R5 - R6` not `R1 - R6`. Furthermore, if path `R1 - R6` is broken, OSPF will find the next shortest path which is `R1 - R3 - R6`. IntServ would install a new reservation along the `R1 - R3 - R6` after failure of path `R1 - R6`, but the best path is `R1 - R2 - R5 - R6`. IntServ has failed to configure and installed QoS for video packets over best path `R1 - R2 - R5 - R6`. This is a major issue since requirements for video applications can vary based on the resolution (*e.g.*, standard-definition (SD), enhanced-definition (ED) or high-definition (HD)).

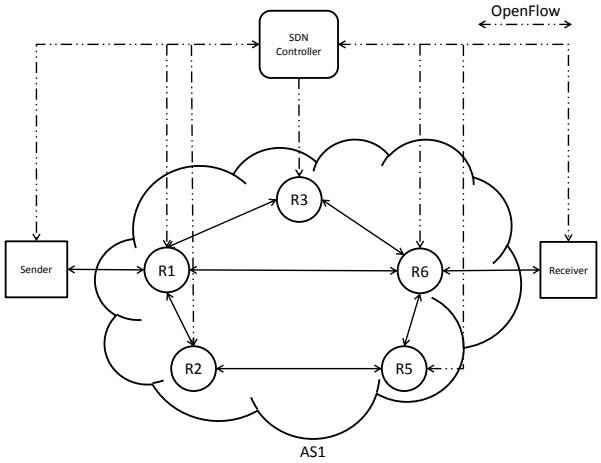

Figure 2: Software-Defined Networking Topology with a Sender and a Receiver

VSDN architecture seen in figure 2 was developed to address path inflexibility limitation of IntServ Architecture. VSDN, like IntServ, is capable of providing QoS guarantees to real-time applications like video. VSDN is capable of selecting not only the shortest path but the optimum path

for video in terms of bandwidth, delay and jitter.

B. Current Usage of VSDN and its limitations

Although VSDN can select the optimum path for video applications using bandwidth, delay and jitter, it is limited in two ways when supporting end-to-end QoS for video applications like telesurgery. First, VSDN only considers a single path when a video application requests service. This is a major limitation because a single path can fail. For example, in figure 2 if video packets are traversing path `R1 - R6` and the path fails, VSDN will need to recognize that a failure has occurred and re-calculate a new path.

Second, VDSN does not consider reliability when making path selections. VSDN should aggregate reliability over multiple paths to ensure QoS for video applications. For example in figure 2, if all paths have reliability of 0.92 and the application request reliability of 0.95 VSDN will have to reject the request or service the request with no guarantee using reliability of 0.92. If VSDN is able to aggregate reliability across two paths (*e.g.*, `R1 - R6` and `R1 - R3 - R6`), it can ensure reliability of 0.9936 (*i.e.*, $(1-(1-0.92)\times(1-0.92)))$ across multiple paths.

Network has become multiple path, mobile devices have multiple radio interfaces, computer devices have multiple network interfaces and data centers have multiple paths [8]. End host video application will need to support multiple path transport to take advantage of the reliability support of RVSDN. Multiple path TCP [8] has been shown to be feasible. A detailed explanation of multi-path transport layer support is outside the scope of this paper.

The network architecture should be able to select the most reliable paths and aggregate reliability across multiple paths. Aggregating reliability across multiple paths allows the network to perform to specification per request. Having the network operator configure explicit paths across links using a protocol like MPLS is *hard*. This paper argues that the network architecture should automatically handle the video application's request of ensuring reliability. The remainder of this paper discusses how Reliable Video over Software-Defined Networking (RVSDN) addresses the issue of reliability.

III. Design and Implementation

This section discusses the design and implementation of Reliable Video over Software-Defined Networking (RVSDN). This section also discusses how RVSDN addresses the challenges introduced in Section II-B.

1) Video Over Software-Defined Networking (VSDN): Figure 3 illustrates the core architecture of VSDN. The links are labeled with QoS constraints (*i.e.*, bandwidth Mbps, delay ms, jitter ms, reliability). VSDN utilizes SDN [9] and OpenFlow [10] to separate the control and forwarding planes of the network devices (*i.e.*, routers and switches). The control plane (*i.e.*, routing) is implemented in the VSDN

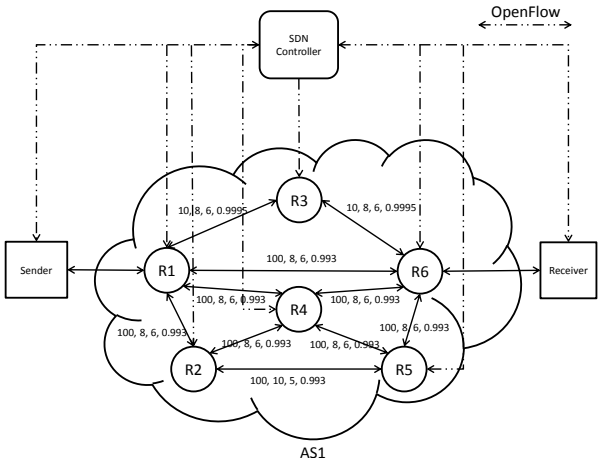

Figure 3: Software-Defined Networking Topology with Link Constraints

controller which resides outside of the forwarding plane as shown in figure 3. The control plane and forwarding plane communicate with one another using OpenFlow. A key component of VSDN is the Routing Module (RM) [5]. The RM which is located in the VSDN controller uses constraint-based routing to calculate feasible paths from ingress router to egress router [5]. Constraint-based routing (CBR) with two or more constraints has been shown to be an NP-hard [11]. Therefore, a heuristics (*i.e.*, A *Prune Algorithm) is used to find a feasible path through the network.

*2) A *Prune Algorithm:* A *Prune algorithm combines A*-search with a correct pruning technique [12]. A *Prune algorithm can be used to solve finding the K shortest paths subject to multiple constraints (KMCSP). A *Prune algorithm takes a graph `G`, with vertices `V` and edges `E`. A *Prune starts at path `P(s, s)` where `s` is a starting vertex in `G`. It expands all possible paths `P(s, V)` that can be reached from `s`. It performs specific pruning against constraint `C`, only the paths in admissible head path set `P(s, V, H(p), C)` are considered. The paths are ordered in a way that the path with the shortest project path length $H_0(p)$ is expanded first. The algorithm terminates when there is a set of constraint shortest path (CSP) found or there are no candidate paths found. There are 7 key processing steps in A *Prune which are combined to select, expand and prune the candidate CSP until the algorithm terminates [12].

A. Changes to Video Over Software-Defined Networking (VSDN) Routing Module

The Routing Module (RM) of VSDN utilizes a variation of the A *Prune algorithm [12] to perform constraint-based routing using bandwidth, delay and jitter as metrics. RVSDN supports the reliability constraint and aggregation of reliability across multiple paths unlike VSDN. A network path supporting video applications like telesurgery may support bandwidth, jitter and delay constraints but if the

network path is not reliable, the performance of the network per request cannot be guaranteed. RVSDN addresses the reliability concern of the network operator automatically by routing based on reliability of network paths.

Algorithm 1 Find Reliable Paths

procedure FINDRELIABLEPATH(G, B, D, J, R)
G: Network Graph
B: Bandwidth
D: Delay
J: Jitter
R: Reliability

$EC = CreateEdgeConstraint(B, D, J, R)$
$R1 = GetIngressSwitch(G)$
$R6 = GetEgressSwitch(G)$
$P = GetFeasiblePath(R1, R6, EC)$
$RP = GetReliablePath(P, R)$

return RP
end procedure

Algorithm 2 Install Reliable Paths

procedure INSTALLRELIABLEPATH($RP, UUID$)
RP: Reliability Path(s)
$UUID$: Unique Path ID

if $(AcquireFlowResource(RP))$ **then**
$PathDatabase.Insert(UUID, RP)$
$InstallReliablePath(OFPFC\_ADD, RP)$
return $TRUE$
end if

return $FALSE$
end procedure

Algorithm 1 illustrates the pseudo code that is implemented in RVSDN controller to support reliable path computation and selection. The user creates an edge constraint `EC` that takes as parameters B, D, J and R where B is bandwidth, D is delay, J is jitter and R is minimal reliability for video application. The ingress and egress routers (*i.e.*, `R1` and `R6` in figure 3) are retrieved using `GetEgressSwitch` and `GetIngressSwitch`. `GetFeasiblePath(R1, R6, EC)` is a core functionality of the routing module (RM). `GetReliablePath(P, R)` takes candidate paths `P` and reliability constraint `R` as parameters. `GetReliablePath(P, R)` sorts the candidate paths. The RVSDN controller installs the paths as illustrated in algorithm 2. The RVSDN controller acquires the resources for reliability paths using `AcquireFlowResource(RP)`. The reliability paths are stored in the path database using a unique id `UUID`. Finally, the RVSDN controller installs the reliability paths using `InstallReliablePath(OFPFC_ADD, RP)`.

For example, assume a video application requests reliability of 0.993 and `FindReliablePath(G, B, J, R)` returns 4 paths with reliability 0.91, 0.75, 0.94 and 0.89. `GetReliabilityPath(P, R)` will sort paths `P` (0.94, 0.91, 0.89, 0.75). `GetReliabilityPath(P, R)` will check to determine if the first path with reliability 0.94 meets reliability constraint. If not, it will calculate the reliability of the first two paths (*i.e.*, $0.94 + 0.91 - 0.94 \times 0.91$) which is 0.9946. A reliability of 0.9946 meets constraint for reliability 0.993. `GetReliablePath(P, R)` will return reliability paths `RP` (*i.e.*, paths with reliability of 0.94 and 0.91). The VSDN controller will update the path database and the flow tables of each OpenFlow switch in reliability paths `RP` after the admission controller (*i.e.*, `AcquireFlowResource(RP)`) determines that resources are available for request.

Algorithms 1 and 2 illustrate the ease of use for controller developers to find constraint-based paths and update the OpenFlow switches. RVSDN and VSDN are fully integrated into the NS3 [7] simulator. This paper discusses results from integrating RSVDN into NS3 in the next section.

IV. RESULTS

In this section, this paper analyzes the number-of-requests serviced by the network architectures based on reliability (*i.e.*, the network architecture ability to perform to specification per request).

A. Experimental Setup

Bandwidth	Delay	Jitter	Reliability
3.0Mbps	150ms	30ms	0.90 - 1.00

Table I: Video Service Request Constraints

Performance metrics - This paper chooses the following performance metric to assess the performance of the RVSDN architecture:

- *Number-of-Requests Serviced by Architecture* - measures the number of requests serviced by network architecture (*i.e.*, VSDN, MPLS and RVSDN). The actual constraints per request are shown in table I.

All experiments were performed on an AMD Athlon X2 5400 system configured with Fedora 18 and 4GB RAM. This system's development environment is typical for developers using NS3 v3.16 [13] and Linux based systems.

B. Experimental Results

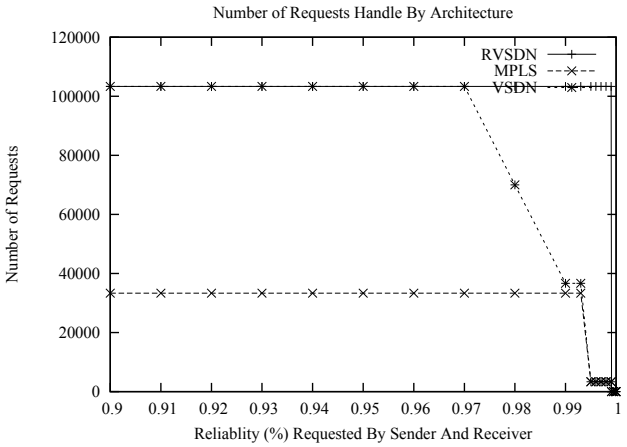

Figure 4: Number of Requests Serviced

Figure 4 illustrates the number of requests serviced by each network architecture using the network topology in figure 3.

MPLS services 33,333 requests before rejecting all other request at reliability 0.90 because MPLS uses path `R1 - R6` which is 100Gbps bandwidth. As stated earlier, MPLS is using explicit routes (*i.e.*, `R1 - R3 - R6` and `R1 - R6` in figure 3). Therefore, after the label switching paths resources are exhausted, MPLS is unable to guarantee quality of service for video requests. VSDN and RVSDN are both able to services 103,333 requests at reliability 0.90 because VSDN and RVSDN can explore undiscovered paths. RVSDN behaves similar to VSDN because there is no need to aggregate links at reliability 0.90. All paths in figure 3 have reliability greater than 0.90 in figure 4. Therefore, as shown in figure 4 each network architecture is able to service end host's requests consistently between reliability 0.90 and 0.97.

MPLS services 33,333 requests at reliability 0.98. Path `R1 - R6` has reliability of 0.993 which meets reliability of 0.98 when using MPLS. VSDN number of requests serviced decreased to 70,000 requests at reliability 0.98 because VSDN does not aggregate reliability across paths. VSDN uses the reliability of a single path. Therefore, only paths `R1 - R3 - R6` (*i.e.*, $0.995 \times 0.995 = 0.990$), `R1 - R6` (*i.e.*, 0.993) and `R1 - R4 - R6` (*i.e.*, $0.993 \times 0.993 = 0.9860$) meet the reliability constraint of 0.98. RVSDN continue to service 103,333 because it has the ability to aggregate reliability over multiple paths at reliability constraint 0.98. VSDN number-of-requests serviced decreases to 36,666 at reliability 0.99 because only paths `R1 - R3 - R6` and `R1 - R6` meet reliability constraint of 0.99. VSDN at reliability 0.99 behaves similar to MPLS because VSDN uses a single path when satisfying the reliability constraint. MPLS is able to service 33,333 requests at reliability constraint 0.99. RVSDN number-of-requests serviced remain constant at reliability 0.99 because it aggregates reliability across links. Each network architecture number-of-requests serviced remains the same at reliability 0.993. VSDN and MPLS number-of-requests serviced drops to 3,333 because both architectures only use path `R1 - R3 - R6` which has bandwidth of 10Gbps and reliability of 0.999 (*i.e.*, $0.999 = 0.9995 \times 0.9995$). RVSDN is able to continue to service 103,333 requests because it aggregates reliability across links which allows it to service more requests at reliability constraint 0.999.

RVSDN's ability to aggregate reliability across links allows it to service more network requests than MPLS and VSDN in terms of number-of-requests serviced. VSDN is able to service more request than MPLS in terms of number-of-requests serviced because VSDN paths are not explicit and can dynamically be discovered.

V. Related Works

Determining the most reliable path (MRP) between two nodes in a network is a well-known problem. Typically, the MRP is determined by using a find shortest path first algorithm similar to Dijkstra or Floyd. Petrovic [14] uses a labeling procedure and a matrix algorithm to compute the MRP. RVSDN uses a variation of the A *Prune Algorithm with a combination of Dijkstra shortest path algorithm. RVSDN like [14] creates an adaptive routing process that is capable of selecting the most reliable path between nodes. RVSDN differs from techniques purposed by Petrovic [14] in that it uses not only the MRP but an aggregation of paths to ensure reliability of service.

Lee et al. [15] select the most reliable path considering the link cost and capacity (*i.e.*, average queue sizes). Lee et al. [15] use random early detection (RED) which is an algorithm for avoiding network congestion using buffer management in routers. Lee et al. [15] use Floyd shortest path algorithm and combines the probability of packets being dropped on link by RED algorithm to select the MRP. RVSDN uses a variation of the A *Prune algorithm combined with Dijkstra shortest path algorithm to calculate the MRP. It was not shown that the algorithm proposed by Lee et al. [15] actually out performs the fixed value operational probability method of finding the MRP. RVSDN does not use the queue length on the links when calculating the MRP. In this case, each router would need to report their average queue length to the RVSDN controller. It has been shown that statistic gathering is an expensive operation on the routers and switches [16]. More research is needed to determine if this method is cost effective. The idea of finding the most reliable path under abnormal traffic conditions purposed by Lee et al. [15] is a technique that RVSDN could use to improve robustness.

Wang et al. [17] use the MRP to ensure the delivery of relief material after a natural disaster. Wang et al. [17] use the concept of detour vital edge to choose the adjustable

reliable path which has higher connectivity reliability and minimal detour distant. Wang et al. [17] present three shortest path algorithms (*i.e.*, depth first search, Dijkstra) and models and compared the modified versions of each algorithm. They use a modified version of Dijkstra shortest path to compute the reliability and weight. Their algorithm is bound by time so it does not run forever. Both traffic and communication networks can be complicated after a natural disaster. Although the networks are complicated, Wang et al. [17] illustrated the feasibility and the correctness of finding the most reliable path after a natural disaster. RVSDN does not assume that a natural disaster has occurred. RVSDN can use the same concepts of detour edge and the ability to handle abnormal traffic conditions [15] to calculate aggregated MRP in the communication network after a natural disaster. Failures were not introduced into the network in this paper.

VI. Conclusion

This paper presented the design and implementation of Reliable Video over Software-Defined Networking (RVSDN). RVSDN builds on previous work [5] of providing end-to-end quality-of-service for video applications and devices that require bandwidth, delay and jitter constraints. RVSDN added the support for routing based on reliability constraint to the VSDN path selection process. RVSDN used multiple paths when determining if network architecture can service requests with reliability constraint. RVSDN was able to service network request that required 0.999 reliability where as MPLS and VSDN ability to service such requests decreased drastically starting at reliability constraint 0.995. Furthermore, RVSDN was able to service 31 times more requests when compared to VSDN and MPLS at reliability constraint 0.995 or greater.

References

[1] Cisco, "Cisco visual networking index: Forecast and methodology, 20122017," http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf, 2012.

[2] R. Chakravorty, S. Kar, and P. Farjami, "End-to-end internet quality of service (qos): An overview of issues, architectures and frameworks," in *Proceedings of International Conference on Information Technology (ICIT)*, 2000.

[3] T. H. Szymanski and D. Gilbert, "Provisioning mission-critical telerobotic control systems over internet backbone networks with essentially-perfect qos," *IEEE J.Sel. A. Commun.*, vol. 28, no. 5, pp. 630–643, jun 2010.

[4] S. Shenker and C. Partridge, "Specification of guaranteed quality of service," http://www.ietf.org/rfc/rfc2212.txt, 1997, rFC 2212.

[5] H. Owens and A. Durresi, "Video over software-defined networking (vsdn)," in *Proceedings of the 16th International Conference on Network-Based Information Systems (NBiS'2013)*, September 2013.

[6] S. Upadhyaya and G. Devi, "Mingling multipath routing with quality of service," *International Journal of Computer Science Issues (IJCSI)*, vol. 8, no. 5, pp. 156 – 161, September 2011.

[7] G. F. Riley and T. R. Henderson, "The ns-3 network simulator modeling and tools for network simulation," in *Modeling and Tools for Network Simulation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, ch. 2, pp. 15–34.

[8] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley, "How hard can it be? designing and implementing a deployable multipath tcp," in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 29–29.

[9] O. N. F. (ONF), "Software-defined networking: The new norm for networks," https://www.opennetworking.org/images/stories/downloads/white-papers/wp-sdn-newnorm.pdf, 2012.

[10] OpenFlow, "Openflow switch specification, version 1.3.0," https://www.opennetworking.org/images/stories/downloads/specification/openflow-spec-v1.3.0.pdf, 2012.

[11] O. Younis and S. Fahmy, "Constraint-based routing in the internet: Basic principles and recent research," *IEEE Communications Surveys and Tutorials*, vol. 5, pp. 2–13, 2003.

[12] G. Liu and K. G. Ramakrishnan, "A*prune: An Algorithm for Finding K Shortest Paths Subject to Multiple Constraints," in *IEEE INFOCOM*, vol. 2, 2001, pp. 743–749.

[13] "The ns-3 network simulator," http://www.nsnam.org/.

[14] R. Petrovic and S. Jovanovic, "Two algorithms for determining the most reliable path of a network," *Reliability, IEEE Transactions on*, vol. R-28, no. 2, pp. 115–119, 1979.

[15] D.-H. Lee, D.-Y. Kim, J.-I. Jung, and Y.-S. An, "An algorithm for acquiring reliable path in abnormal traffic condition," in *Proceedings of the 2008 International Conference on Convergence and Hybrid Information Technology*, ser. ICHIT '08. IEEE Computer Society, 2008, pp. 682–686.

[16] J. C. Mogul and P. Congdon, "Hey, you darned counters! get off my asic!" in *SIGCOMM '12: Proceedings of the ACM SIGCOMM 2012 conference on data communication*. ACM, Aug. 2012, pp. 3–14.

[17] J. Wang, J. Zhu, and H. Yang, "Reliable path selection problem in uncertain traffic network after natural disaster," *Mathematical Problems in Engineering*, vol. 2013, 2013, 5 pages.